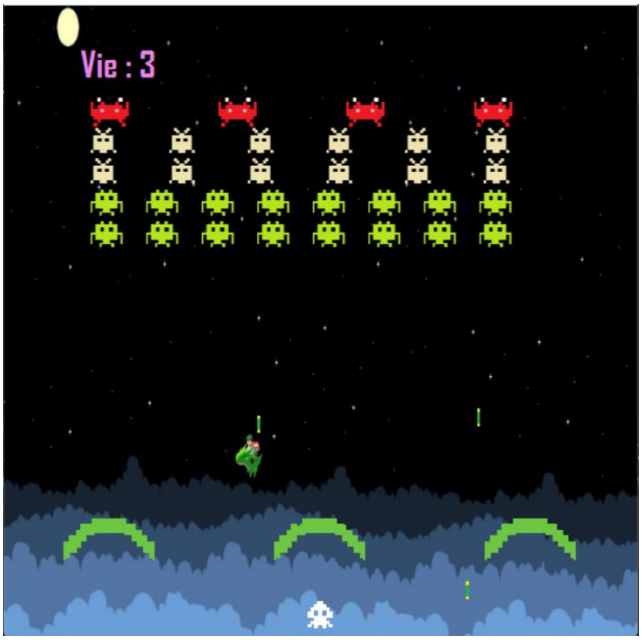
22/12/2023

Rapport du Projet :

Space Invaders



Réalisé Par :

DEHMANI Manar & MOUHAMAD Afouane

1. Introduction

1.1.Contexte et objectif du projet

Dans le cadre du projet dans le module POO nous avons reproduis un classique du jeu vidéo, un Space Invaders.

Space Invaders est un jeu vidéo développé par la société japonaise Taito en 1978 sur borne d'arcade. Il s'agit d'un shoot'em up fixe. Le principe du jeu est de détruire des vagues d'aliens au moyen d'un canon en se déplaçant horizontalement sur l'écran.

Pour la réalisation de ce projet, en tant que binôme de débutants en programmation orientée objet, surtout en C#, nous avons délibérément choisi la piste verte. Cette décision vise à nous familiariser avec les principes fondamentaux de la programmation orientée objet.

1.2.Cahier des charges

Pour assurer le bon déroulement de notre projet Space Invaders, nous établissons un cahier des charges détaillé. Les éléments fondamentaux définissent les composants essentiels qui constitueront notre interface de jeu. Voici les spécifications clés :

A. Interface Principale :

- Le vaisseau du joueur est au cœur de l'interface, représentant le protagoniste de l'aventure.
- Une rangée de bunkers constitue une première ligne de défense pour le joueur.
- Les vaisseaux ennemis, disposés en blocs et en lignes, seront les cibles à éliminer.
- La présence de missiles, à la fois pour le joueur et les ennemis, ajoute une dimension stratégique au gameplay.
- Les sprites/bitmap seront utilisés pour visualiser les vaisseaux, bunkers et missiles à l'écran.

B. Déplacement du Joueur :

- Le joueur peut déplacer son vaisseau horizontalement à l'aide des touches fléchées gauche et droite.
- Le déplacement du vaisseau se fait uniquement dans le plan horizontal, et il ne peut pas sortir de l'écran.

C. Organisation des Ennemis :

- Les ennemis sont agencés en blocs, formant plusieurs lignes où chaque ligne représente un type de vaisseau spécifique.

- Le bloc d'ennemis se déplace horizontalement de gauche à droite, et lorsqu'il atteint le bord de l'écran, il se décale vers le bas.
- À chaque changement de direction, la vitesse de déplacement du bloc augmente, intensifiant le défi.

D. Mouvement des Missiles :

- Les missiles ennemis se déplacent verticalement, dirigés vers le bas.
- Les missiles du joueur se déplacent également verticalement, mais la direction dépend du vaisseau d'origine.

Ce cahier des charges posera les bases nécessaires à la mise en œuvre de Space Invaders, en s'assurant que chaque élément contribue de manière significative à l'expérience de jeu.

2. Description du Problème et Structure du Programme

1.1. Description succincte du problème

Le problème auquel nous faisons face consiste à recréer le jeu SpaceInvaders dans un environnement de programmation orientée objet en utilisant le langage C#. L'objectif est de développer un jeu interactif où le joueur peut contrôler un vaisseau pour détruire des vagues d'ennemis. Le gameplay implique des éléments tels que le déplacement du joueur, mis en place des bunkers, le mouvement des ennemis, le tir de missiles alliés et ennemies, et les interactions entre ces composants.

Space Invaders se distingue par sa dynamique de déplacement des ennemis en blocs, la nécessité de détruire les vaisseaux ennemis tout en évitant les missiles adverses, et l'ajout de fonctionnalités telles que la vitesse croissante des ennemis et la défense des bunkers. L'objectif est de créer une expérience de jeu fluide et divertissante qui respecte les règles et caractéristiques originales du jeu classique.

2.2. Structure du programme

Durant ce projet, nous étions amenés à utiliser les fondamentaux de la programmation orienté objet entre autres : les classes, l'héritage etc...

Par conséquent, nous générer à l'aide de Visual Studio la structure de notre programme sous forme d'un diagramme de classe qui explique la hiérarchie entre les classes (les liens d'héritage en les classes) et qui montrent les méthodes de chaque classe.

Mais avant ceci nous allons définie les principales classes de notre jeu SpaceInvaders.

- GameObjects: La classe mère du programme

- Game : La classe qui représente le jeu où il y'aura tout l'ajout les objets crées avec les différentes méthodes nécessaires pour le jeu.

- EnemyBlock: c'est la classe qui représente le bloc ennemi du jeu et qui hérite de GameObject.

- SimpleObject: c'est la classe mère de SpaceShip , missile et bunker et en même temps elle hérite de GameObjects. Cette classe représente toutes les fonctionnalités communes entre ses classes filles.

-**Bunker :** c'est la classe qui représente les bunkers permettent au joueur de se mettre à couvert mais peuvent être détruit par les missiles. Les bunkers ne se déplacent pas.

-Missile : c'est la classe qui représente les tirs du vaisseau joueur ainsi que les tirs du block ennemies et qui hérite de SimpleObject.

-SpaceShip: Cette classe hérite de SimpleObject, et en même temps c'est la classe mère de PlayerSpaceShip , elle gère tous les vaisseaux (joueur et ennemies).

-PlayerSpaceShip: C'est la classe qui représente le vaisseau joueur

-Bonus : C'est la classe qui représente le bonus qui tombe avec une certaine probabilité quand le vaisseau ennemi meurt.

-SoundManager : La classe qui gère le son du jeu.

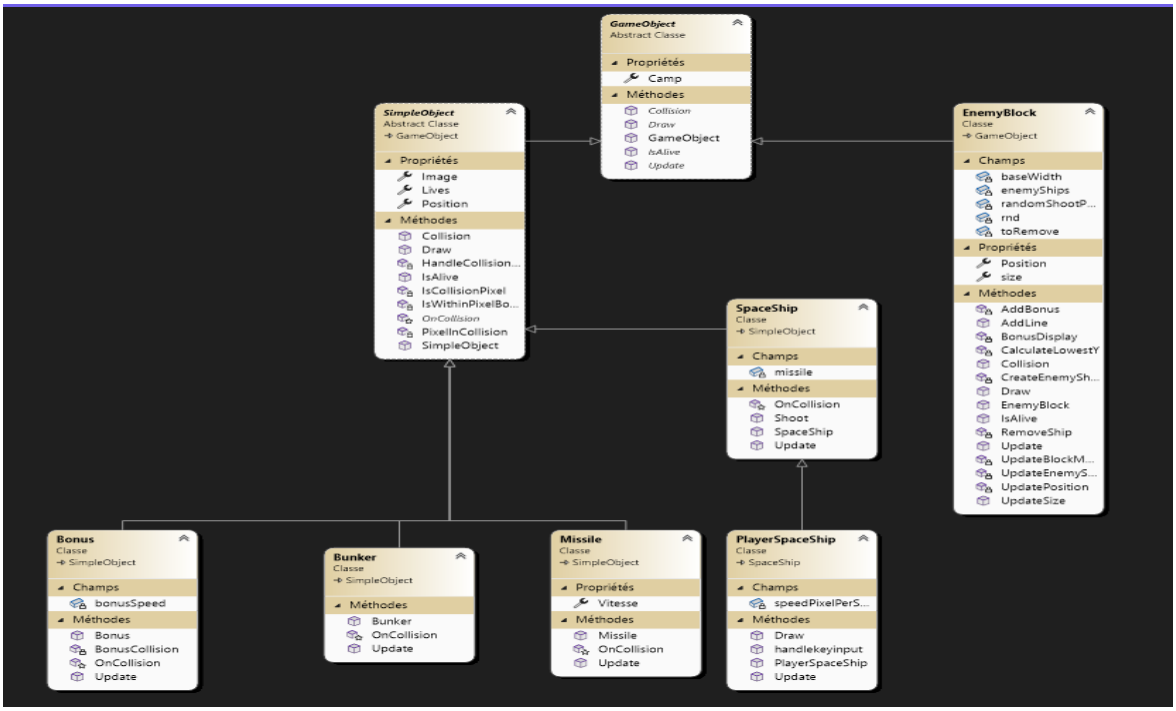


Illustration n°1 : Structure du projet n°1

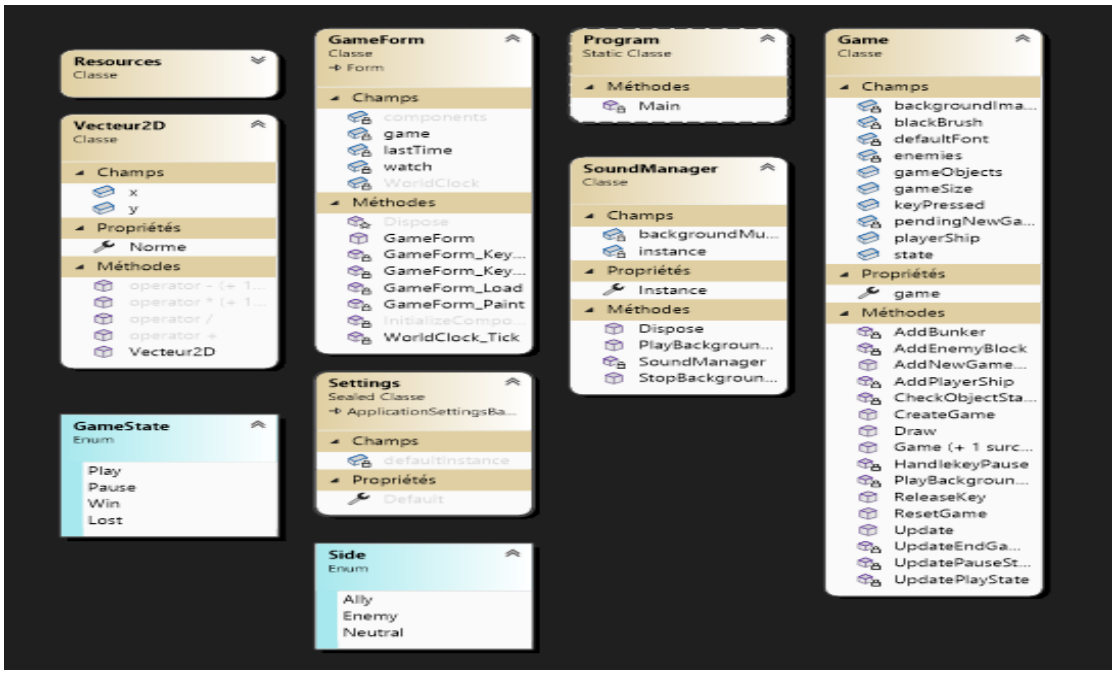


Illustration n°2 : Structure du projet n°2

3. Problèmes Rencontrés et Solutions Apportées

3.1. Analyse des problèmes rencontrés lors du développement

a. GitHub : Nous avons rencontré au début un problème avec l'utilisation du git. Si l'un de nous push alors que le deuxième a modifié les mêmes classes nous avions des conflits qui se déclenche et nous n'arrivons pas à résoudre les conflits engendrés vu que nous n'étions pas familiarisés avec l'utilisation du git.

b. PlayerSpaceShip: Au début nous n'arrivions pas à afficher le vaisseau joueur, même si nous avons ajouté tout ce qui était demandé

c. Collisions : La mise en place de cette fonctionnalité était un peu complexe à mettre en place au début avec la difficulté des différents calculs.

d. EnemyBlock: Une autre difficulté a été rencontrée lors du développement du jeu, et cela en ajoutant les vagues des vaisseaux ennemies en prenant compte les autres vagues ajoutés, au début les vagues s'empilent à la même position.

3.2. Solutions apportées ou idées de solutions envisagées.

a. Pendant, la première période d'entreprise nous étions amenés à travailler avec git, nous avons fait une petite formation sur GitHub et par conséquent ça nous a aidé pour notre projet et nous avons réussi à résoudre tout problèmes engendrés avec GitHub.

b. Après avoir compris la logique et la première structure du jeu, nous avons bien compris qu'il fallait appeler la méthode Draw du vaisseau joueur dans la classe Game.

c. Collisions : Comme nous l'indique l'énoncé nous avons créé les sous fonctions pour une meilleur gestion et compréhension de cette fonctionnalité et concernant les calculs de pixel nous avons comparé les positions des bunkers et des missiles.
d. Pour résoudre ce problème, nous avons choisis une variable qui était la coordonné y la plus basse, qui représente la position de la ligne ennemie la plus basse et nous avons ajouté une valeur fixe pour créer un intervalle entre deux vagues successives.

4. Addons

4.1.Son

Dans le cadre de l'expérience immersive de notre projet Space Invaders, nous avons accordé une attention particulière à l'aspect sonore. Une ambiance sonore de fond a été intégrée pour plonger le joueur au cœur de l'univers du jeu. Pour ce faire, nous avons délibérément opté pour un son emblématique tiré du célèbre jeu "Geometry Dash". Cette référence familière a été incorporée pour susciter un sentiment de nostalgie tout en apportant une touche distinctive à notre adaptation de Space Invaders.On note que la classe SoundManager utilise la classe SoundPlayer pour contrôler la musique de fond du jeu.

4.2. Bonus

Nous avons introduit un élément de jeu supplémentaire sous la forme de bonus. Ces bonus, générés aléatoirement après la destruction d'un vaisseau ennemi se déplaçant du haut vers le bas. Ils offrent une opportunité stratégique au joueur. Un bonus spécifique, lorsqu'attrapé par le vaisseau joueur, accorde une vie supplémentaire, augmentant ainsi la durée de jeu.

4.3.Design

Pour rendre le jeu beaucoup plus attirant, et pour sortir du noir et blanc nous avons pensé d'ajouter un background image pour notre jeu ainsi nous avons changé les couleurs des bunkers, des vaisseaux joueurs voire le vaisseau joueur, pour une meilleure visualisation. Nous étions amenés ainsi de modifier la couleur au sein de la méthode IsCollisionPixel afin de garder toujours le principe de la collision.

5. Conclusion et Perspectives

5.1.Conclusion

En guise de conclusion, il parait clair que ce projet avait un rôle primordial surtout pour nous en tant que débutants en programmation orienté objet notamment en C#, car grâce à ce projet nous avons pu développer nos compétences en ce langage de programmation et atteindre notre objectif fixé.

5.2.Perspectives d'amélioration

Par la suite, nous pourrions envisager d'ajouter plusieurs sons afin d'offrir au joueur une meilleure expérience, tels que le bruit de l'explosion des vaisseaux ennemis, le tir de missiles, le son de la victoire, etc. Nous pourrions également intégrer un menu de démarrage

comprenant des options telles que "Jouer", "Aide" et "Quitter". De plus, il serait intéressant d'implémenter un système de niveaux avec une gestion des scores, permettant aux joueurs de se confronter les uns aux autres.